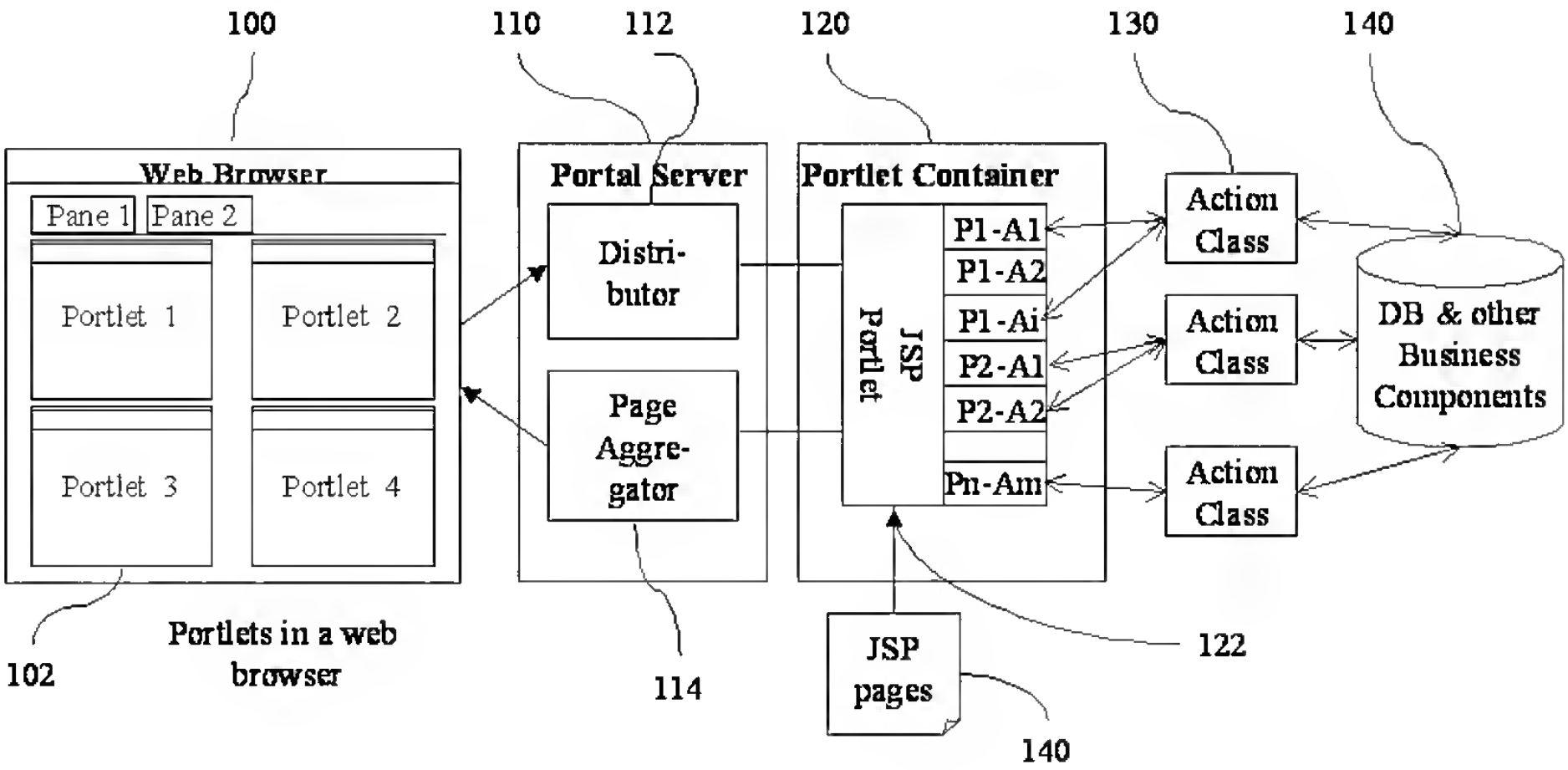


FIG. 1: Prior Art

200

```
<%@ taglib uri='/WEB-INF/templates/jsp/tld/template.tld' prefix='jetspeed' %>
<%@ page import = "org.apache.turbine.util. *" %>
<%@ page import="org.apache.jetspeed.modules.actions.portlets.JspPortletAction" %>
<%@ page import="org.apache.jetspeed.portal.Portlet" %>
<%@ page import="org.apache.jetspeed.util.PortletSessionState" %>

<%try {
        RunData rundata = (RunData) request.getAttribute("rundata");
        final String userid = rundata.getUser().getUserName();
        String jspeid = (String) request.getAttribute("js_peid");
        String address = (String) request.getParameters.getAttribute("address");
} %>
<FORM name="addressFrm"  method="post" action="<jetspeed:dynamicUri/>">
        <INPUT TYPE="hidden" NAME="js_peid" VALUE="<%=jspeid%>">
        <textarea name="address"><%= address %></textarea>
        <input type="submit" name="eventSubmit_doUpdate" value="Submit">
</form>
<% }
catch (Exception e) {
        AppObjects.log("Error:error in jsppage",e);
        return;
}%>
```

FIG. 2: Prior Art

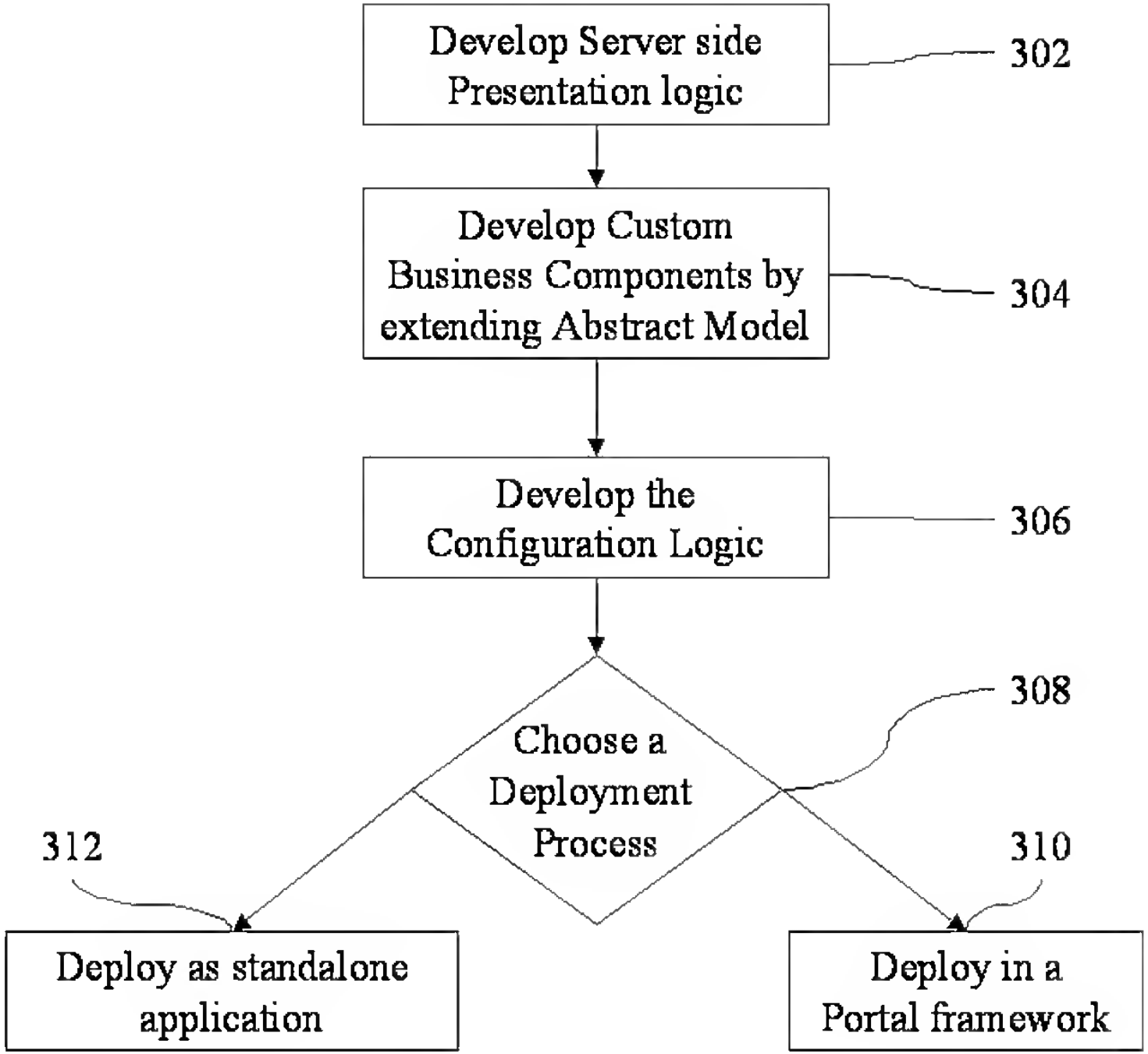

Develop Server side Presentation logic
302
Develop Custom Business Components by extending Abstract Model
304
Develop the Configuration Logic
306
Choose a Deployment Process
308
312
310
Deploy as standalone application
Deploy in a Portal framework

FIG. 3

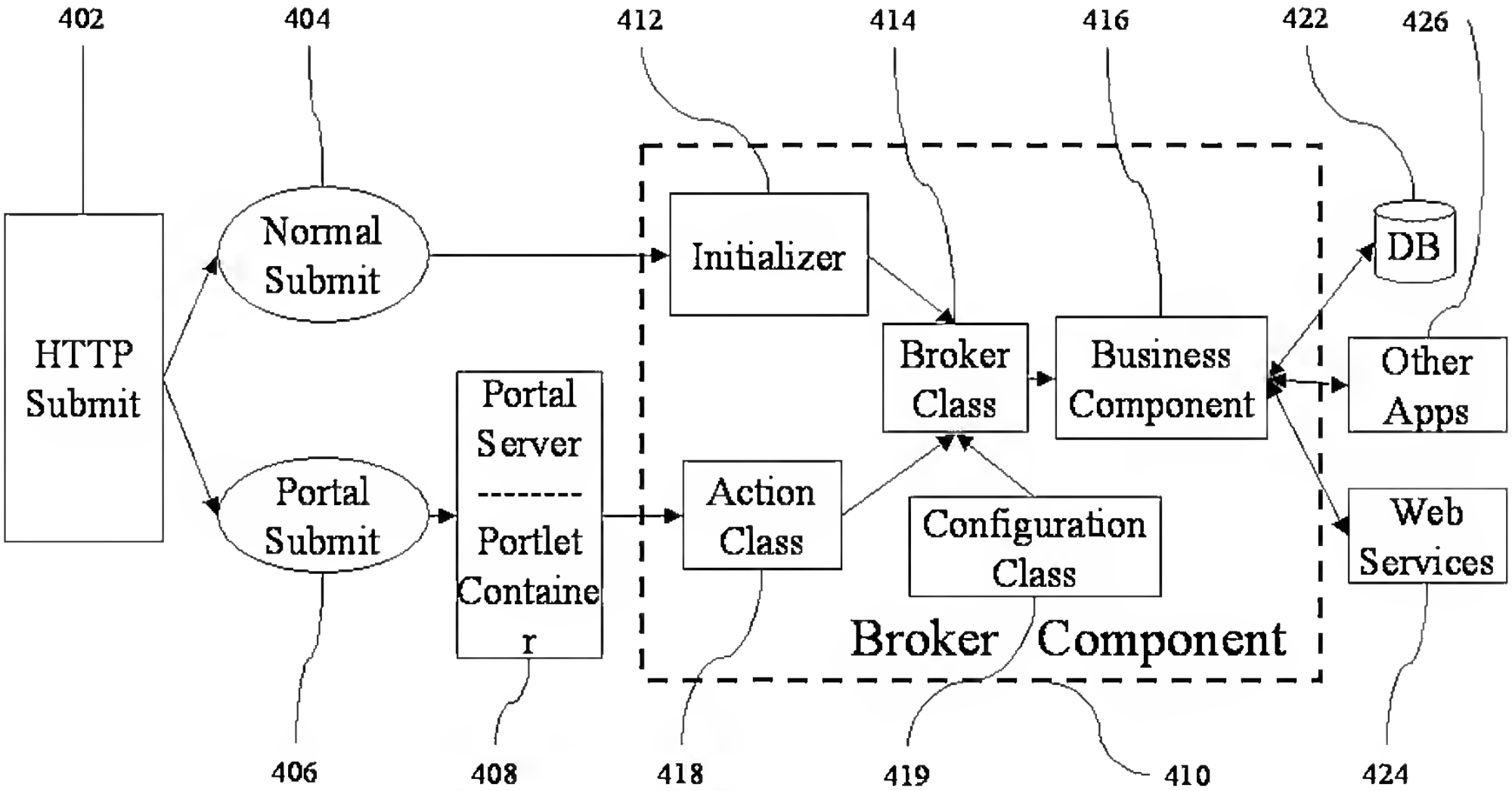
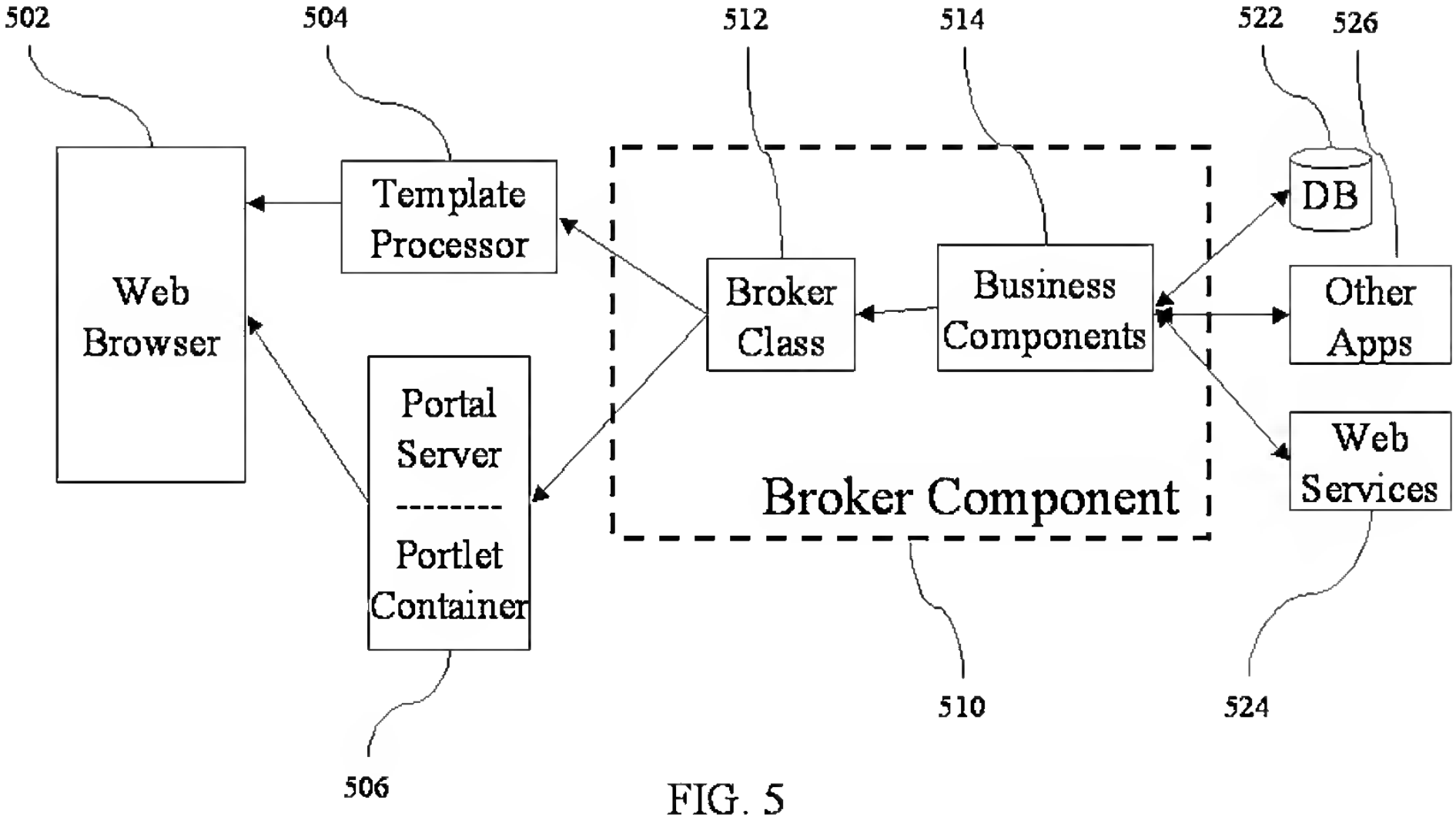

402
404
412
414
416
422
426
HTTP Submit
Normal Submit
Portal Submit
Portal Server
Portlet Container
Initializer
Action Class
Broker Class
Business Component
Configuration Class
DB
Other Apps
Web Services
Broker Component
406
408
418
419
410
424

FIG. 4


502
504
512
514
522
526
Web Browser
Template Processor
Broker Class
Business Components
DB
Other Apps
Portal Server

Portlet Container
Broker Component
Web Services
510
524
506

FIG. 5

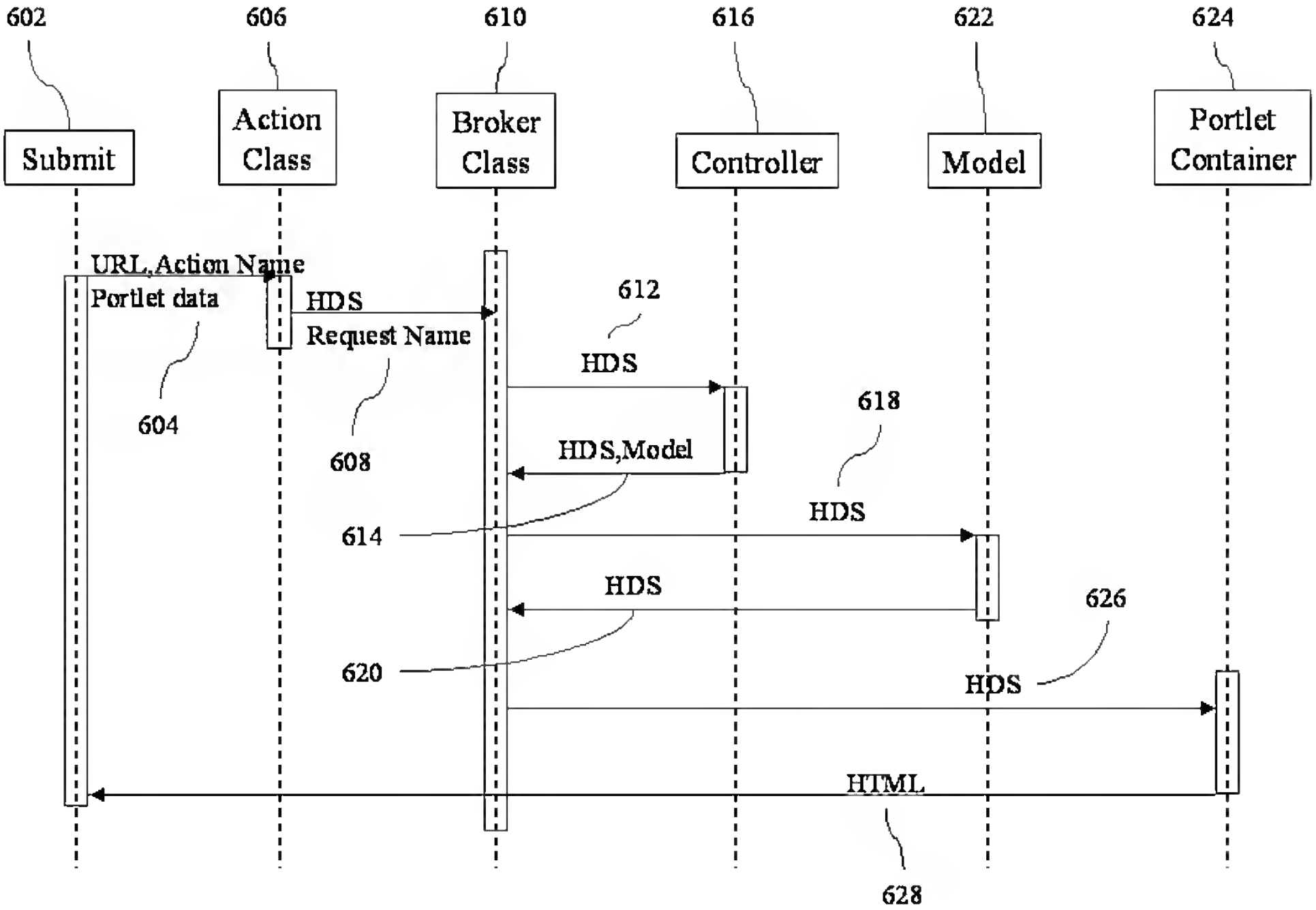

602
606
610
616
622
624
Submit
Action Class
Broker Class
Controller
Model
Portlet Container
URL,Action Name
Portlet data
HDS
Request Name
612
HDS
604
608
618
HDS,Model
HDS
614
HDS
626
620
HDS
HTML
628

FIG. 6

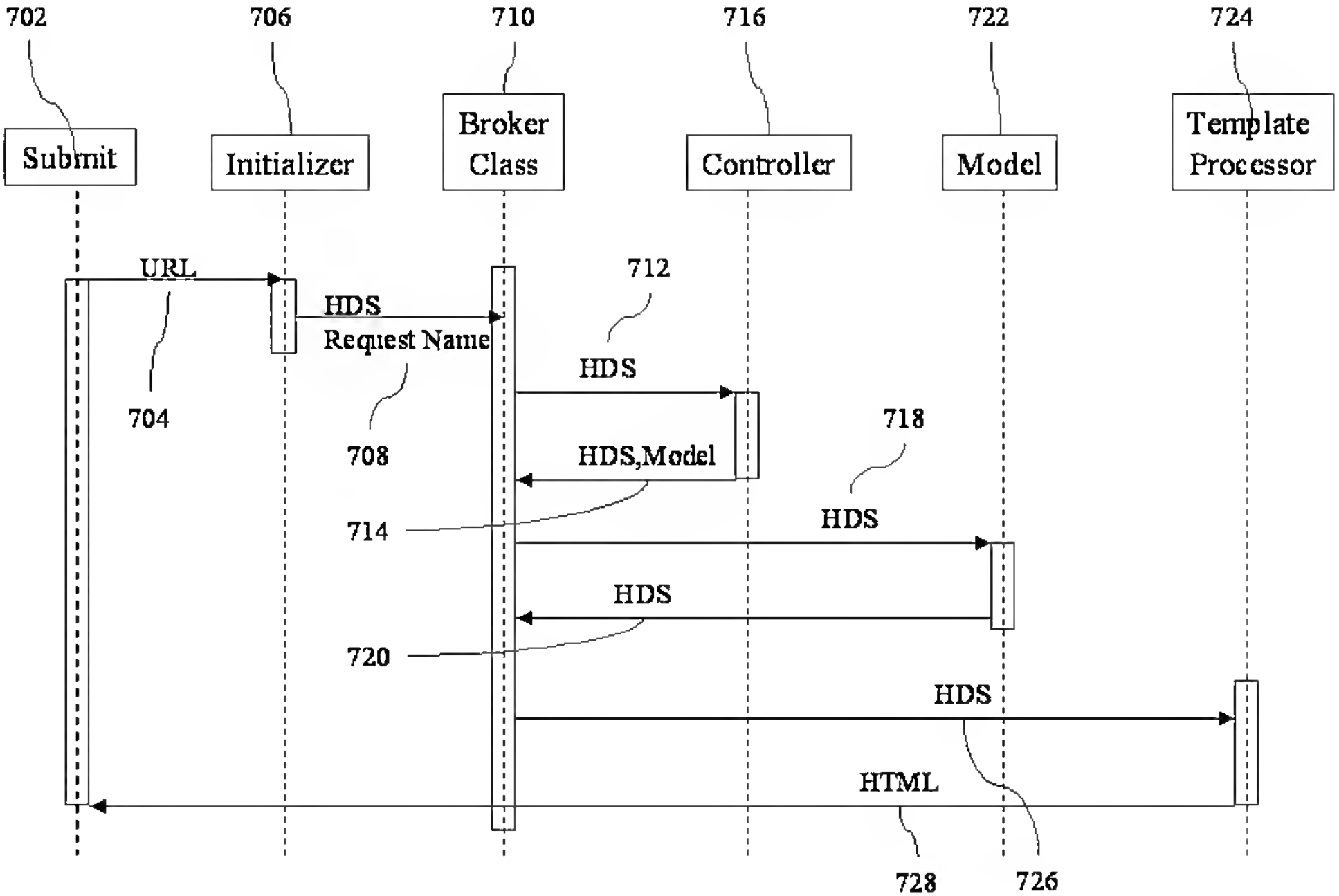

702
706
710
716
722
724
Submit
Initializer
Broker Class
Controller
Model
Template Processor
URL
704
HDS
Request Name
708
712
HDS
HDS,Model
718
HDS
714
HDS
720
HDS
HTML
728
726

FIG. 7

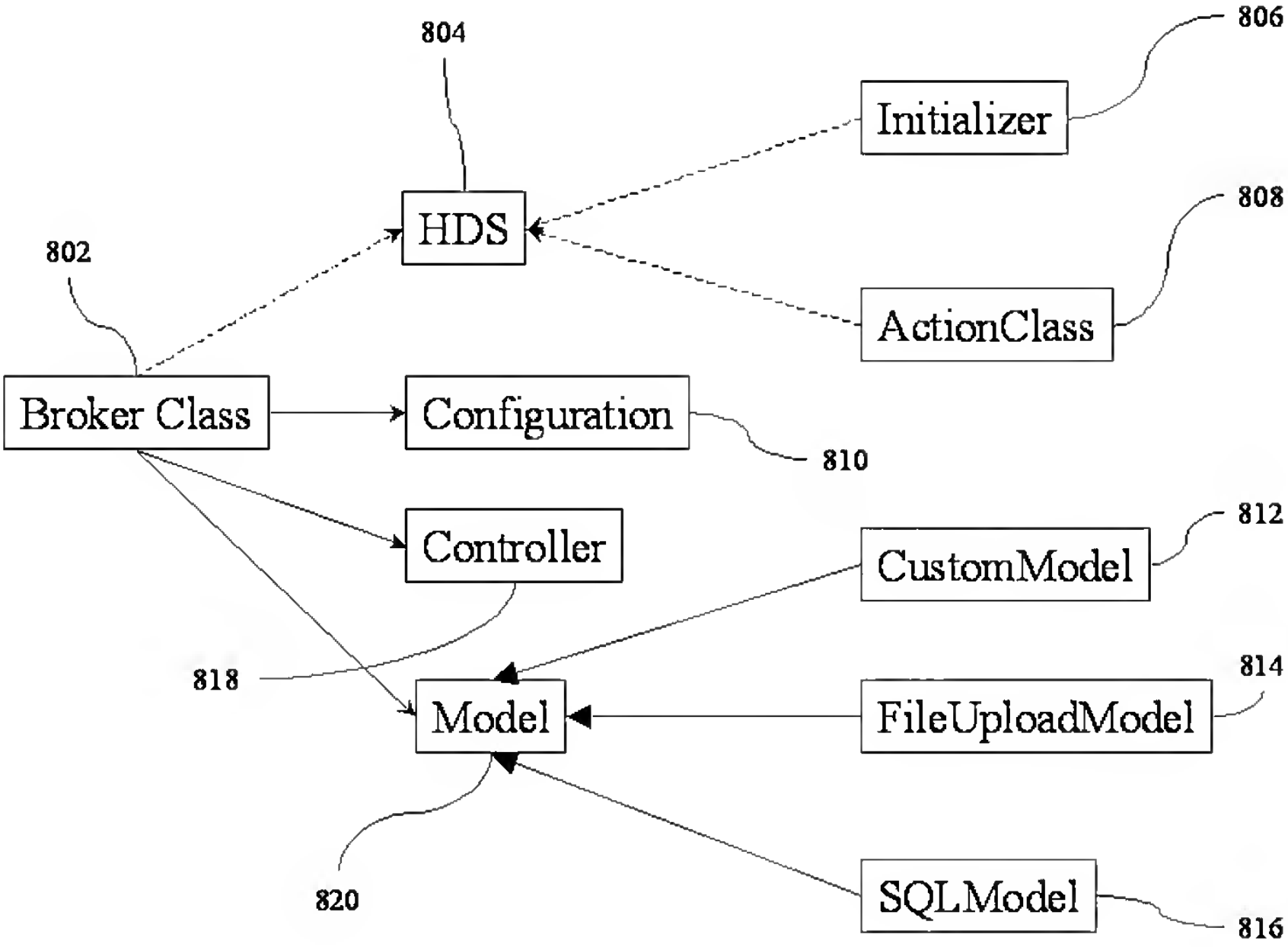

804
806
Initializer
HDS
808
802
ActionClass
Broker Class
Configuration
810
Controller
812
CustomModel
818
814
Model
FileUploadModel
820
SQLModel
816

FIG. 8

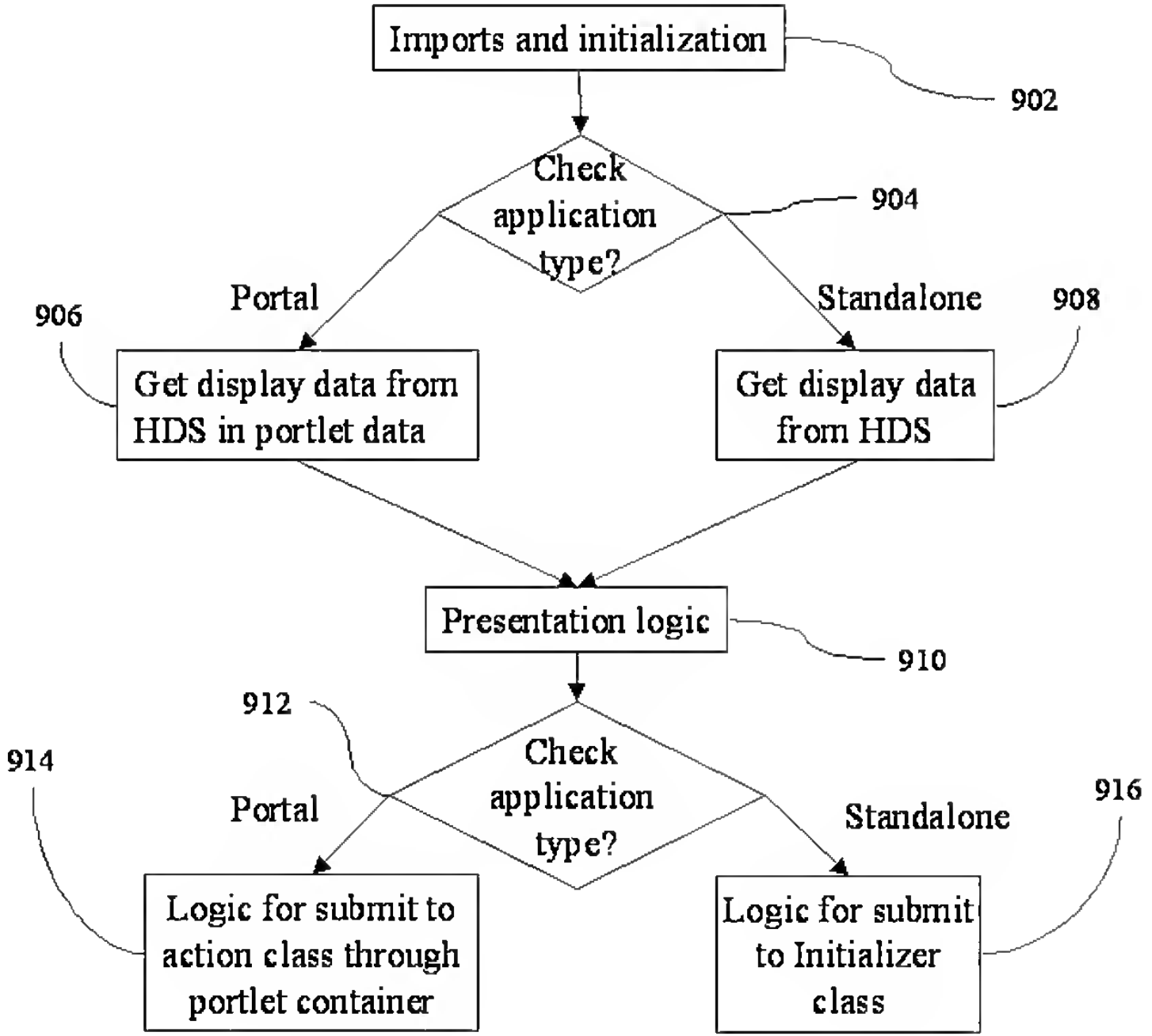

Imports and initialization
902
Check application type?
904
Portal
Standalone
906
908
Get display data from HDS in portlet data
Get display data from HDS
Presentation logic
910
912
Check application type?
914
Portal
Standalone
916
Logic for submit to action class through portlet container
Logic for submit to Initializer class

FIG. 9

1002

```
Class Broker {
    Model model;
    Controller controller;
    Configuration configuration;
    ……
    public Broker() {
        configuration =Registry.getConfiguration();
        controller = new Controller(configuration);
    }
    public void executeRequest (String requestName) {
        ArrayList arrayActions = controller.getActionArray(requestName);
        String template = controller.getTemplate(requestName);
        for(int i=0;i< arrayActions.length();i++) {
            executeModel(arrayActions [i]);
        }
        setTemplateProcessor(template );
    }
    public void executeModel (String actionName) {
        Object model = controller.getModel(actionName);
        model.execute();
    }
    public void setTemplateProcessor(String template) {
        Registry.getHDS().setTemplate(Registry.getTemplate(template));
    }
}
```

FIG. 10

1102

```
Class ActionClass {
    String requestName=null;
    ……
    public ActionClass () {
    }
    protected void buildNormalContext(Portlet portlet, RunData rundata) {
        Registry.getHDS().setParams(portlet.getPortletConfig().getInitParameters());
        Registry.getHDS().setParams(rundata.getParameters());
        callBroker(Registry.getHDS().getRequest());
    }
    public void callBroker(String request) {
        Registry.getBroker().executeRequest(request);
    }
}
```

FIG. 11

1202

```
abstract class Model {
    abstract void execute() {
    }
}
```

1204

```
public class SQLModel extends Model {
  public Connection con =null;
  public String actionName = null;
  public SQLModel(String action) {
      actionName = action;
      try{ Class.forName("hds.getDriver()");
        con = DriverManager.getConnection(hds.getDB(),hds.getUser(), hds.getPassword());
      } catch(Exception e) {}
  }
  public void execute() {
      stmt = con.createStatement();
      ResultSet rs = null;
      try { rs = stmt.executeQuery(Registry.getSQL(actionName));
        Registry.updateHDS(rs);
      }catch(Exception e) {}
      finally{rs=null; con =null;}
  }
}
```

FIG. 12